Efficient Web Requests Scheduling Considering Resources Sharing

Simon Duquennoy, Gilles Grimaud

Efficient Web Requests Scheduling Considering Resources Sharing

Simon Duquennoy
LIFL, CNRS UMR 8022, Univ. Lille 1,
INRIA Lille - Nord Europe, France
Email: simon.duquennoy@lifl.fr

Gilles Grimaud
LIFL, CNRS UMR 8022, Univ. Lille 1,
INRIA Lille - Nord Europe, France
Email: gilles.grimaud@lifl.fr

***Abstract*—Requests scheduling in Web servers is a hot research topic. Many works aim at providing optimal algorithms according to various metrics. Most of these works are based on classical scheduling metrics, considering jobs completion times, but ignoring intermediate states. We claim that this choice conduces to the design of algorithm that do not efficiently share the system resources. Indeed, Web servers have some properties that make them different than the system considered in usual scheduling theory. The classical round-robin policy, used in most production Web servers, has intrinsic qualities: it shares equally the system resources and avoids any job starvation. We introduce a novel parameterizable algorithm proposing a compromise between the benefits of the round-robin and the policies that provide the best performances. Then, we discuss the appropriate choice of the parameter depending in the requirements and the context of the Web server.**

I. Introduction

Web servers have to handle many simultaneous client requests, so they have to share their resources (network link, CPU, memory) as efficiently as possible. The scheduling of requests in Web servers is often analyzed as a queuing problem; client requests are considered as input jobs, server responses are considered as job service. Many theoretical results analyze scheduling algorithms in regards to various metrics in order to evaluate their performances and/or fairness.

Widely used Web servers (*e.g.* Apache, IIS) are implemented using the sockets provided by the underlying OS, which does not allow to control the TCP level scheduling. The simple and fair Processor Sharing (PS) policy approximates the behavior of a scheduler using the classical round-robin.

It has been proved [1] the SRPT policy optimizes the mean flow-time (or response time) for any input. The common assumption made in the works promoting the usage of SRPT is that job intermediate state are irrelevant. Only completion times are considered. We claim that this assumption biases the results in the particular context of Web requests, in which intermediate states do matter.

The motivation behind this work is that we are convinced that the simple and widely used PS algorithm has very good properties in the context of Web requests scheduling. By construction, it shares the system resources fairly. By serving all requests at the same rate, every client gets service at any time, and there is no request starvation. Starting from this observation, it is temping to improve the very efficient SRPT by considering both performances and resources sharing.

A. Optimizing performances

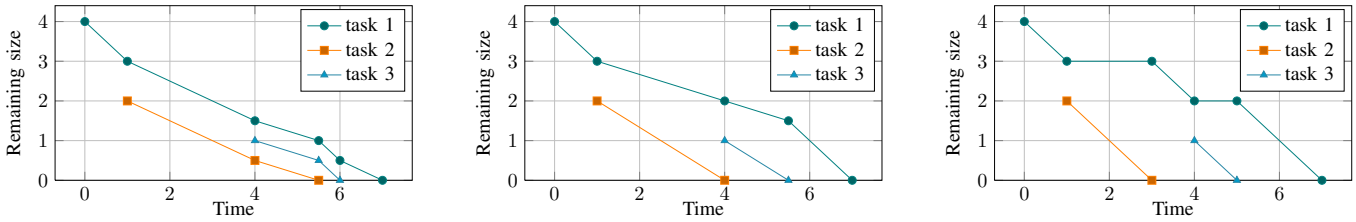
The Shortest Remaining Processing Time (SRPT) policy has been proved to minimize the mean flow time [1]. With this policy, the scheduler devotes its full attention at any time to the job with the shortest remaining processing time. In the case of Web requests, the remaining processing time is the remaining size of the HTTP response to be sent. It has however been shown (proved [2], simulated [3] and experimented [4]) that under realistic load (M/G/1 heavy-tailed queue), SRPT only slightly penalizes long tasks in comparison to PS. The long tasks penalization only appears under heavy loads. For certain sample paths, every task performs better or equally under SRPT than under PS.

B. Considering resources sharing

Several works focus on the classification of scheduling algorithms according to their fairness properties. In [5], the notions of endogenous and exogenous unfairness are defined. In [6], algorithms are considered as fair if they do not favor jobs depending on their sizes. In [7], an algorithm is said to be fair *iff* it provides a response time that is less or equal than under PS for every task and any sample path. Following this definition, SRPT is showed to be unfair. A new protocol called Fair Sojourn Protocol (FSP) is presented, approaching SRPT performances while guaranteeing fairness. Like SRPT, FSP prioritizes tasks with short remaining processing time, but it avoids long jobs starvation. In [8], two hybrid algorithms allowing to smoothly join SRPT and PS are presented. In all of these works, intermediate job state are not considered: only completion time does matter.

In [9], the authors propose a new measure of fairness considering both jobs seniority and size, called RAQFM. It is based on the notion of individual discrimination, calculated as the integration, during the job sojourn time, of the difference between the resources granted to the job and the warranted resources (inverse of the number of jobs). Let d_i be the departure time of job i, $s_i(t)$ be the fraction of the resources allocated to job i at time t and $N(t)$ the number of jobs in the system at time t. The discrimination D_i is defined as follows:

$$D_i = \int_{a_i}^{d_i} \left(s_i(t) - \frac{1}{N(t)} \right) dt$$



(a) $\beta = 1$ (PS). Completions at 7, 5.5 and 6. (b) $\beta = 0.5$. Completions at 7, 4 and 5.5. (c) $\beta = 0$ (SRPT). Completions at 7, 3 and 5.

Fig. 1. Behavior of β-SRPT with various β values for a same sample path

The authors measure the fairness of a schedule by computing the variance of jobs individual discriminations. They formally calculate the expected RAQFM of various standard scheduling algorithms such as FIFO, LIFO, PS, *etc.* By construction, RAQFM considers intermediate job states.

II. Discussion of the metrics in the context of Web requests scheduling

Solutions allowing existing Web server like Apache to use SRPT have been presented, requiring modifications inside the operating systems in order to adapt sockets management. However, Web server still do not integrate policies like SRPT or FSP, preferring the classical PS policy. We claim that this gap between theory and practice is due to the usage of inappropriate metrics when analyzing scheduling algorithm in the special case of Web requests scheduling. We identify four main reasons why the mean flow-time is not an appropriate metric in this case. These four points also explain why the classical PS policy is still preferred by the industry to SRPT:

- In practice, sharing a link with multiple hosts is far more efficient than bursting clients consecutively (because of network congestions and TCP congestion control) ;
- The fact of keeping a link idle a long time involves unnecessary overheads: TCP keep-alive messages (void acknowledgements), connections timeouts (involved by proxies or firewalls) *etc.* ;
- Web contents are designed to be displayable gradually, so requests intermediate state do matter. Once the beginning of a page has been received, the browser starts running the Web application, requests nested resources, *etc.* ;
- Completely stopping the service of certain requests to serve others do not provide a good user experience: after waiting a few seconds, the user of a Web application will conclude that the request will never be served and he may trigger a new request, thus increasing the server load.

In spite of these drawbacks, the flow-time metric presents a major quality in the case of Web servers. By optimizing the mean flow-time, SRPT also minimizes the number of jobs in the system at any time, increasing the server scalability. This reduction of the number of current requests allows to improve the server pure performances by improving file cache sizes and accelerating system calls which execution time increase linearly with the number of sockets (such as `select()`).

For these reasons, we claim that the flow-time metric should be used carefully in the context of Web requests scheduling. It should be completed with a metric evaluating the instantaneous fairness, thus allowing to fairly share the system resources and considering requests intermediate states. That is why we propose to evaluate scheduling algorithms by combining both mean flow-time and RAQFM metrics.

The optimization of the RAQFM instantaneous fairness is trivially obtained with the PS algorithm: by sharing equally the available resources at any time, the individual discrimination of every job equals 0. Intuitively, SRPT does not provide a good RAQFM, since it tries to minimize resources sharing and does not consider intermediate states. On the other hand, PS does not provide a good mean flow-time, while SRPT optimizes this metric. In the next section, we investigate compromises between SRPT and PS for both performances (measured with the mean-flow time) and instantaneous fairness (measured with RAQFM).

III. Balancing performances and resources sharing

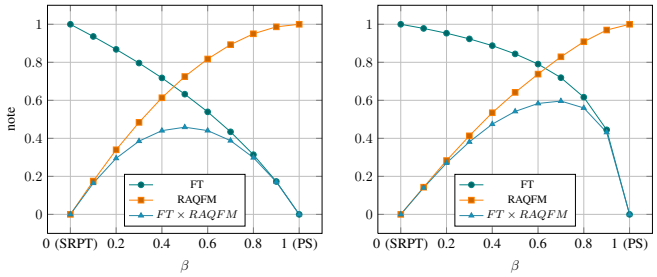
We propose a novel scheduling algorithm, β-SRPT, which aim is to balance the efficiency of SRPT with the way PS fairly shares the system resources. The trade-off can be parametrized with the β value, which can be chosen depending on the requirements of the Web server.

A. Algorithm description

The β-SRPT algorithm smoothly joins SRPT and PS. With $\beta = 0$, the algorithm behaves like SRPT, while with $\beta = 1$, it behaves like PS. Intermediate values for β give compromises between both algorithms. β-SRPT is based on a General PS algorithm, which schedules jobs simultaneously with a given weight, depending on β. At any time, the N jobs in the system are sorted by increasing order of remaining processing time (noted R_j). The weight w_j for job j is defined as follows:

$$w_j = \frac{\beta^j}{\sum_{i=0}^{N-1} \beta^i} = \begin{cases} \frac{1}{N} & \beta = 1 \\ \beta^j \times \frac{1-\beta}{1-\beta^N} & \beta \neq 1 \end{cases}$$

In other words, the jobs priority decreases exponentially with the rank (ordered by increasing remaining processing time) of the job. With $\beta = 1$, every job has the same weight $\frac{1}{n}$, so that the policy is strictly equivalent to PS. With $\beta = 0$, we have $w_0 = 1$ and $w_j = 0$ for any $j > 0$. Since jobs are



(a) System load: $\rho = 0.5$ (b) System load: $\rho = 0.9$

Fig. 2. Flow-time/Instantaneous fairness compromise with β-SRPT

ranked by order of increasing remaining processing time, this policy is strictly equivalent to SRPT.

Depending on the β value, the β-SRPT algorithm lies between SRPT and PS, both in terms of flow-time and resources sharing. Whatever the β value, the $j+1^{th}$ job is served β time slower than the j^{th} job. The closer β is to 0, the most efficient will be the algorithm in terms of flow-time, since it favors jobs with the shortest remaining processing time. The closer β is to 1, the most fairly are the weights assigned to the jobs, thus sharing the resources equally. Figure 1 illustrates the behavior of β-SRPT for various β values.

Note that since β-SRPT is based on a General PS algorithm, it considers time slots to be indefinitely divisible. For implementation, a packet oriented version of the algorithm is needed, and can be precisely obtained with the Weighted Fair Queuing policy [10].

B. First simulation results

We evaluate the behavior of β-SRPT by simulating it. Following empirical analysis [11], it is widely accepted that Web traffics can be precisely modeled using a M/G/1 heavy-tailed queue. The inter-arrival process follows a Poisson distribution of rate λ. The job sizes distribution is a bounded Pareto. As in [2], we use a shape $\alpha = 1.1$ for the bounded Pareto jobs size distribution. The minimal job size x_{min} is chosen such that the mean job size is fixed at 1 and the maximal job size is 10^5. The system load ρ is calculated as the product between the inter-arrival rate and the mean job size. Every run has been done with 1 million jobs in the system.

Our simulations confirmed that the β-SRPT algorithm provides intermediate solutions between SRPT and PS, both in terms of performances (flow-time and number of jobs in the system) and fairness (lack of endogenous unfairness and RAQFM note). When using β-SRPT, one has to choose the appropriate β value by answering some questions. Do the clients need short response times? Do they need fairness? Is the Web server memory-constrained? Is it heavily loaded?

In order to evaluate the compromise reached depending on β, we give to every schedule a normalized note between 0 and 1 for the mean flow-time and for the RAQFM instantaneous fairness. The SRPT and PS policies are used to calibrate the notes. Figure 2 shows the notes obtained depending on β. The product of both notes is also given, synthesizing the trade-off.

The crossing between the two curves identifies the β reaching an equilibrium between flow-time and instantaneous fairness. We notice that this crossing is not encountered for the same β depending on the system load. As an example, in an heavy loaded system ($\rho = 0.9$), if one needs a scheduler that shares almost equally the link (close to PS) but slightly prioritizes short requests in order to improve response times, he can choose $\beta = 0.9$. The RAQFM note is 0.97, and the flow-time note is 0.44, resulting in an interesting trade-off.

IV. Conclusions & Future Works

The motivation behind this work comes from the fact that the classical flow-time metric is not sufficient to evaluate Web requests scheduling algorithms; it has to be completed with a measure of the way system resources are shared. We proposed the parameterizable β-SRPT algorithm allowing to smoothly join the properties of PS and SRPT respectively in terms of instantaneous fairness and flow-time. Our simulations shows that efficient compromises are possible, where the gain on one metric is more important than the loss on the other one.

As future works, we plan to implement the β-SRPT algorithm in order to evaluate it in real context. We also would like to study the dynamic adaptation of the β value at runtime. As an example, one could set a threshold on the minimal instantaneous fairness to be reached, letting the β-SRPT algorithm set the appropriate β parameter, which may vary with the system load, the incoming traffic and the properties of the network paths to clients. Our approach could also be extended to consider individual job weights, representing the memory consumed by every job in the system.

References

[1] L. E. Schrage, "A proof of the optimality of the shortest processing remaining time discipline," *Operations Research*, vol. 16, p. 678, 1968.

[2] N. Bansal and M. Harchol-Balter, "Analysis of srpt scheduling: Investigating unfairness," 2001, pp. 279–290.

[3] M. Gong and C. Williamson, "Quantifying the properties of srpt scheduling," in *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003*, vol. 0. Los Alamitos, CA, USA: IEEE, oct 2003, pp. 126–135.

[4] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, "Size-based scheduling to improve web performance," *ACM Trans. Comput. Syst.*, vol. 21, no. 2, pp. 207–233, 2003.

[5] M. Gong and C. Williamson, "Revisiting unfairness in web server scheduling," *Comput. Netw.*, vol. 50, no. 13, pp. 2183–2203, 2006.

[6] A. Wierman, "Fairness and classifications," *SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 4, pp. 4–12, 2007.

[7] E. J. Friedman and S. G. Henderson, "Fairness and efficiency in web server protocols," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 229–237, 2003.

[8] M. Gong and C. Williamson, "Simulation evaluation of hybrid srpt scheduling policies," in *The IEEE's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004.*, oct 2004, pp. 355–363.

[9] D. Raz, H. Levy, and B. Avi-Itzhak, "A resource-allocation queueing fairness measure," in *SIGMETRICS '04/Performance '04: Proceedings of the joint international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2004, pp. 130–141.

[10] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, 1993.

[11] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 835–846, 1997.